8-Ч

УДК 519.713.1

А.Н. Чеботарев, А.Л. Головинский
Институт кибернетики им. В.М. Глушкова НАН Украины, г. Киев
ancheb@gmail.com, golovinsky.andriy@gmail.com

Доказательное проектирование алгоритмов функционирования реактивных систем

Описывается подход к доказательному проектированию реактивных алгоритмов, развиваемый в Институте кибернетики имени В.М. Глушкова НАН Украины. Рассматриваются основные проблемы, возникающие при проектировании реактивных алгоритмов, специфицированных в логическом языке *L*, и методы их решения.

Введение

Под реактивными системами понимаются системы, постоянно взаимодействующие со своим окружением. Это, как правило, системы реального времени, работа которых заключается в выработке реакции на изменяющуюся входную информацию. Примерами реактивных систем могут служить телекоммуникационные сети, системы управления технологическими процессами, системы управления летательными аппаратами и др.

Ошибки в функционировании таких систем, как системы управления ядерными реакторами, химическим производством, средствами космической техники могут привести к катастрофическим последствиям, связанным с большими материальными потерями и человеческими жертвами. Поэтому к надежности и безошибочности функционирования такого рода систем предъявляются чрезвычайно высокие требования. В частности, необходимо, чтобы алгоритм функционирования системы точно соответствовал требованиям, определяемым его исходной спецификацией. Этого можно достичь применением строгих методов корректного проектирования алгоритмов функционирования реактивных систем (реактивных алгоритмов), причем большое значение имеет обеспечение корректности алгоритма на начальных этапах его разработки, поскольку стоимость устранения ошибок, допущенных на этих этапах, наиболее велика.

В методах корректного проектирования реактивных алгоритмов, используемых на начальных этапах их разработки, можно выделить два основных подхода:

а) формальная верификация неформально полученного процедурного представления алгоритма,

б) использование математически обоснованных формальных методов для преобразования декларативной спецификации требований к функционированию алгоритма в высокоуровневое императивное (процедурное) его представление.

При первом подходе доказывается, что полученный алгоритм обладает некоторыми необходимыми свойствами, но не гарантируется, что поведение алгоритма будет в точности соответствовать его назначению. При втором подходе, называемом *синтезом,* гарантируется точное соответствие между исходной спецификацией алгоритма и ее процедурной реализацией. В этом случае исходная спецификация должна отражать все требования к функциональному поведению проектируемого алгоритма.

Указанные подходы имеют ряд общих черт. Как при первом, так и при втором подходах для задания исходной спецификации алгоритма или его свойств используются декларативные языки. Как правило, это логические исчисления, позволяющие явно или неявно вводить понятие времени в спецификацию алгоритма. Некоторые методы верификации реактивных алгоритмов, в частности, методы проверки выполнимости спецификации на модели (model checking), так же как и методы второго подхода, используют процедуру синтеза автомата по логической спецификации свойств алгоритма, однако характер решаемых задач, их размерность и методы решения в этих двух подходах существенно различаются. Так, при первом подходе решается задача синтеза автомата-распознавателя, а при втором – автомата-преобразователя, что обусловливает различие языков спецификации. Кроме того, размерность задачи синтеза при втором подходе на несколько порядков превосходит размерность аналогичной задачи при первом подходе. Поэтому методы синтеза автоматов, используемые при верификации, не могут использоваться в рамках второго подхода при проектировании реальных алгоритмов.

Синтез алгоритма – это не однократный акт, а последовательность преобразований, осуществляемых, как правило, с участием человека. При этом результат синтеза алгоритма будет гарантированно удовлетворять исходной спецификации лишь в том случае, если корректность каждого преобразования, осуществляемого в процессе синтеза, будет формально доказана. Это касается не только преобразований, выполняемых автоматически (корректность которых обеспечивается формальным доказательством корректности соответствующих методов), но и осуществляемых с участием человека. Таким образом, всякое вмешательство человека в процесс проектирования должно контролироваться системой проектирования, отвергающей ошибочные действия разработчика. Такую методологию проектирования реактивных алгоритмов будем называть *доказательным проектированием*.

В настоящей статье описывается подход к доказательному проектированию реактивных алгоритмов, развиваемый в Институте кибернетики имени В.М. Глушкова НАН Украины, рассматриваются основные проблемы, возникающие при проектировании реактивных алгоритмов, специфицированных в языке L, и методы их решения.

Язык спецификации

В основе рассматриваемого подхода к доказательному проектированию алгоритма лежит спецификация функциональных требований к нему в языке логики первого порядка с одноместными предикатами и формальный переход от этой спецификации к процедурному представлению алгоритма. Специфицируемый алгоритм представляется в виде двух частей: управляющей и операционной, взаимодействующих между собой и с внешней средой. Взаимодействие между управляющей и операционной частями осуществляется через двоичные каналы. Взаимодействие с внешней средой осуществляется как через информационные каналы, так и через двоичные каналы. Логическая спецификация алгоритма определяет описание его управляющей части и тех аспектов операционной части и внешней среды, которые относятся к взаимодействию управляющей и операционной частей между собой и с внешней средой. Таким образом, логическая спецификация алгоритма состоит из трех частей: спецификаций управляющей части, операционной части и внешней среды. В качестве математической модели каждой из перечисленных частей спецификации рассматривается конечный автомат. Составляющие части алгоритма удобно специфицировать как неинициальные системы, отдельно задавая начальное условие, определяющее начальные состояния соответствующих автоматов.

Один из основных вопросов, связанных с проблемой спецификации свойств алгоритма, состоит в выборе или разработке подходящего языка спецификации.

Идея использования логического языка для спецификации конечных автоматов восходит к работам Черча [1], Бюхи [2] и Б.А. Трахтенброта [3], [4]. В этих работах построен ряд языков, основанных на логике одноместных предикатов от натурального аргумента, и приведены соответствующие алгоритмы синтеза автоматов. Однако полученные в то время результаты не стимулировали применения логических методов для спецификации и проектирования программ или схем, поскольку носили сугубо теоретический характер. В настоящее время существует большое количество языков для спецификации реактивных алгоритмов. К ним относятся языки, основанные на различных темпоральных логиках, наибольшее распространение из которых получили LTL [5], CTL [6], CTL* [7], а также языки, основанные на операторах наименьшей и наибольшей неподвижной точки [8]. Эти языки, как правило, используются в системах верификаии, однако сложность решения для них задач синтеза такова, что вряд ли можно надеяться на применение их для решения практических задач. Существенного увеличения эффективности алгоритмов синтеза можно добиться путем разумного ограничения выразительных возможностей языка спецификации. При этом приходится искать компромисс между выразительными возможностями языка и сложностью алгоритмов проектирования. Для разрешения этого противоречия предлагается использовать два уровня языка: язык исходной спецификации L^* [9], обладающий достаточными для практических нужд выразительными возможностями и обеспечивающий удобство записи исходных требований к алгоритму, и внутренний язык L [10], обладающий сравнительно ограниченными выразительными возможностями, но эффективно обрабатываемый процедурами синтеза. Отметим, что название «внутренний язык» довольно условное, поскольку во многих случаях он может выступать в качестве языка исходной спецификации. Оба языка построены на основе соответствующих фрагментов логики предикатов первого порядка с одноместными предикатами, определенными на множестве моментов дискретного времени, в качестве которого выступает множество Z целых чисел. При этом имеется возможность спецификацию в языке L^* преобразовать в спецификацию в языке L.

Спецификация в языке L имеет вид $\forall t F(t)$, где $F(t)$ – формула, построенная с помощью логических связок из атомов вида $p(t + k)$. Здесь p – предикатный символ, t – переменная, принимающая значения из множества Z, а k – целочисленная константа (сдвиг во времени), называемая *рангом* соответствующего атома. Язык L^* отличается от языка L тем, что при построении $F(t)$ используется еще конструкция: $\exists t_i (t_i \le t + k_1)$ & $F_1(t_i)$ & $\forall t_j((t_i + k_2 \le t_j \le t + k_3) \to F_2(t_j))$, где $k_1, k_2, k_3 \in$ Z, $F_2(t_j)$ – формула языка L, а $F_1(t_i)$ – формула языка L^*.

При определении автоматной семантики языков спецификации эти языки и автоматы рассматриваются как формализмы для задания множеств сверхслов (бесконечных слов) в алфавите Σ двоичных векторов. Этот алфавит определяется набором предикатных символов языка спецификации, которые, в свою очередь, соответствуют входным и выходным двоичным каналам синтезируемого автомата. Определим необходимые понятия, касающиеся сверхслов и автоматов.

Пусть Σ – конечный алфавит, **Z** – множество целых чисел и $\mathbf{N}^+ = \{z \in \mathbf{Z} \mid z > 0\}$. Отображения u: $\mathbf{Z} \to \Sigma$ и $l: \mathbf{N}^+ \to \Sigma$ называются соответственно *двусторонним сверхсловом* (обозначается $\ldots u(-2)u(-1)u(0)u(1)u(2)\ldots$) и *сверхсловом* (обозначается $l(1)l(2)\ldots$) в алфавите Σ. Для двустороннего сверхслова u и $n \in \mathbf{Z}$ определим n-суффикс $u(n+1, \infty)$ как сверхслово $u(n+1)u(n+2)\ldots$

Конечный неинициальный X–Y-автомат Мили A – это четверка $A = <X, Y, Q, \delta>$, где X, Y, Q – конечные множества соответственно входных символов, выходных символов и состояний, а $\delta: Q \times X \times Y \to 2^Q$ – функция переходов автомата.

X–Y-автомат Мили называется *квазидетерминированным*, если для любых $q \in Q$, $x \in X$, $y \in Y$ $|\delta(q, x, y)| \leq 1$. Квазидетерминированные X–Y-автоматы удобно рассматривать как детерминированные частичные автоматы без выхода, с входным алфавитом $\Sigma = X \times Y$. Такой автомат $A = <\Sigma, Q, \delta>$, где δ: $Q \times \Sigma \to Q$ – частичная функция, будем называть Σ-автоматом.

Σ-автомат $A = <\Sigma, Q, \delta>$ называется *циклическим*, если для каждого $q \in Q$, существуют такие $\sigma_1, \sigma_2 \in \Sigma$ и $q_1, q_2 \in Q$, что $q_1 \in \delta(q, \sigma_1)$ и $q \in \delta(q_2, \sigma_2)$.

Циклический автомат может быть однозначно охарактеризован в терминах допустимых сверхслов.

Сверхслово $l = \sigma_1, \sigma_2, \ldots$ в алфавите Σ *допустимо в состоянии* q автомата A, если существует такое сверхслово состояний $q_0 q_1 q_2 \ldots$, где $q_0 = q$, что для любого $i = 0, 1, 2, \ldots$ $q_{i+1} = \delta(q_i, \sigma_{i+1})$. Сверхслово *$l$ допустимо для автомата A*, если оно допустимо в каком-либо из его состояний.

Перейдем теперь к описанию сверхсловарной семантики языка L. Каждой замкнутой формуле F ставится в соответствие множество моделей для этой формулы, т.е. множество таких интерпретаций, на которых F истинна. Пусть $\Omega = \{p_1, \ldots, p_m\}$ – множество всех одноместных предикатных символов, встречающихся в формуле F. Интерпретация формулы F – это упорядоченный набор определенных на $\mathbf{Z}$ одноместных предикатов $\pi_1, \ldots, \pi_m$, соответствующих предикатным символам из Ω. Пусть Σ – множество всех двоичных векторов длины m, тогда интерпретацию $I = <\pi_1, \ldots, \pi_m>$, можно представить в виде двустороннего сверхслова в алфавите Σ, а множество всех моделей для F – в виде множества M_F двусторонних сверхслов в этом алфавите. Будем полагать, что формула $F = \forall t F(t)$ языка L задает множество всех 0-суффиксов двусторонних сверхслов из M_F.

Теорема 1 [11]. Для всякой непротиворечивой формулы F вида $\forall t F(t)$ существует в общем случае частичный неинициальный циклический автомат A, для которого множество всех допустимых сверхслов совпадает с множеством сверхслов, задаваемых формулой F.

Это определяет автоматную семантику языка L.

Проверка непротиворечивости

Проверка непротиворечивости как исходной спецификации, так и различных промежуточных спецификаций, получаемых в процессе синтеза алгоритма, имеет принципиальное значение для методологии доказательного проектирования. Соответствующая процедура используется в процессе проектирования не только для проверки внутренней непротиворечивости спецификации, но также для преобразования ее к виду, необходимому для применения других алгоритмов проектирования. Проверка непротиворечивости используется также для проверки корректности решений (изменений спецификации), принимаемых разработчиком при интерактивном процессе проектирования. Поэтому необходимы достаточно эффективные методы решения этой проблемы.

Для проверки непротиворечивости исходной спецификации используются резолюционные методы логического вывода [12-14], эффективность которых удалось существенно повысить за счет учета специфики предметной области.

Формулу, получающуюся из $F(t)$ путем прибавления константы k к рангам всех ее атомов, обозначим $F(t + k)$. Поскольку формулы интерпретируются на множестве целых чисел, для любого $k \in \mathbf{Z}$ имеет место эквивалентность $\forall tF(t) \Leftrightarrow \forall tF(t + k)$. Это позволяет ограничиться рассмотрением только таких формул, в которых максимальный ранг атомов равен нулю (*нормализованные вправо* формулы).

Пусть $F(t)$ задана в виде к.н.ф., все элементарные дизъюнкции которой нормализованы вправо. Как обычно, элементарную дизъюнкцию литер будем называть *дизъюнктом*. Дизъюнкт, не содержащий литер, называется *пустым*. К.н.ф. формулы $F(t)$ будем задавать в виде множества дизъюнктов.

Пусть c_1 и c_2 – нормализованные вправо дизъюнкты, $p(t)$ – атом нулевого ранга и $c_1 = c'_1 \vee p(t)$, а $c_2 = c'_2 \vee \neg p(t)$. Дизъюнкт $c = c'_1 \vee c'_2$ называется *R-резольвентой* дизъюнктов c_1 и c_2 по атому $p(t)$.

Операцию получения R-резольвенты двух дизъюнктов будем называть R-*резольвированием*.

R-выводом дизъюнкта c из множества дизъюнктов C называется такая конечная последовательность дизъюнктов $c_1, ..., c_k$, что $c_k = c$ и каждый дизъюнкт c_i ($i = 1, ..., k$) либо принадлежит C, либо является R-резольвентой двух предшествующих дизъюнктов, либо есть результат нормализации вправо дизъюнкта c_{i-1}.

Справедливо следующее утверждение. Множество C нормализованных вправо дизъюнктов противоречиво тогда и только тогда, когда существует R-вывод пустого дизъюнкта из C.

Процесс проверки непротиворечивости множества дизъюнктов C можно представить в виде поочередного выполнения следующих двух операций:

а) построение замыкания C относительно резольвирования по атомам нулевого ранга,

б) нормализация вправо всех ненормализованных дизъюнктов.

Процесс проверки выполнимости формулы заканчивается, если после очередного выполнения операции а) полученное множество дизъюнктов будет содержать пустой дизъюнкт, что свидетельствует о противоречивости формулы $\forall tF(t)$, либо новые дизъюнкты не появятся, все дизъюнкты будут нормализованы вправо и полученное множество дизъюнктов не содержит пустого дизъюнкта. Вторая альтернатива имеет место в случае выполнимости формулы.

Дизъюнкт $c_1(t)$ *поглощает* дизъюнкт $c_2(t)$, если существует такое $k \in \mathbf{Z}$, что все литеры из $c_1(t+ k)$ содержатся среди литер дизъюнкта $c_2(t)$. Удаление из множества дизъюнктов, представляющих формулу $\forall tF(t)$, дизъюнктов, поглощаемых другими дизъюнктами этого множества, приводит к множеству дизъюнктов, задающих формулу, эквивалентную (т.е. имеющую то же множество моделей в рассматриваемом классе интерпретаций) формуле $\forall tF(t)$.

Описанная выше процедура может быть усовершенствована за счет удаления в процессе ее выполнения дизъюнктов, поглощаемых другими дизъюнктами, принадлежащими преобразуемому множеству. Усовершенствованную таким образом процедуру будем называть *пополнением* множества дизъюнктов.

Теорема 2 [12]**.** Формула $\forall tF(t)$, заданная множеством C нормализованных вправо дизъюнктов, невыполнима тогда и только тогда, когда в процессе пополнения C будет получен пустой дизъюнкт.

Эффективность рассмотренного метода обусловлена сокращением множества порождаемых дизъюнктов за счет ограничения вида литер, по которым допускается резольвирование, литерами нулевого ранга. При этом отпадает необходимость выполнения унификации, что также повышает эффективность метода. В [13], [14]

описаны усовершенствования рассмотренного метода, связанные с разбиением множества дизъюнктов на несколько классов и запрещением резольвирования дизъюнктов, принадлежащих различным классам, что существенно сокращает количество дизъюнктов, порождаемых в процессе пополнения.

Синтез

Методы синтеза управляющей части алгоритма основаны на теореме о спецификации [9], устанавливающей связь между структурой некоторого представления формулы в языке спецификации и структурой соответствующего автомата, определенной в терминах множества состояний и функций переходов и выходов. Это приводит к понятию нормальной формы формулы $F(t)$ языка L.

Пусть $F(t) = \bigvee_{i=1}^{n} F_i(t-1)\&f_i(t)$, где $F_i(t-1)$ – формула, максимальный ранг атомов в которой не превышает -1, а $f_i(t)$ – формула, построенная из атомов ранга 0. Представление $F(t)$ в таком виде будем называть *дизъюнктивным представлением*, а конъюнкцию вида $F_i(t-1)\&f_i(t)$ – *компонентой* такого представления с *левой частью* $F_i(t-1)$ и *правой частью* $f_i(t)$. Дизъюнктивное представление удовлетворяет *условию ортогональности*, если для $i \neq j$ $(i, j \in \{1, \ldots, n\})$ $F_i(t-1)\&F_j(t-1) \equiv 0$. Дизъюнктивное представление формулы $F(t)$, удовлетворяющее условию ортогональности, называется *нормальной формой*, если для любых $i, j = 1, \ldots, n$ конъюнкция $F_i(t-1)\&f_i(t)\&F_j(t)$ либо тождественно равна нулю, либо равна $F_i(t-1)\&f_{ij}(t)$, где $f_{ij}(t)$ – не равная тождественно нулю формула, построенная из атомов нулевого ранга.

Пусть $\Omega = \{p_1, \ldots, p_q\}$ – множество всех предикатных символов, встречающихся в $F(t)$, а $\Sigma(\Omega)$ – множество всех двоичных векторов длины q. Символу $\sigma \in \Sigma(\Omega)$ соответствует элементарная конъюнкция $\tilde{\sigma}(t)$ вида $\tilde{p}_1(t)\& \ldots \&\tilde{p}_q(t)$, где $\tilde{p}_j(t) \in \in \{p_j(t), \neg p_j(t)\}$. Спецификации $F = \forall tF(t)$, в которой $F(t)$ представлена в нормальной форме, соответствует Σ-автомат $A(F)$ с входным алфавитом $\Sigma(\Omega)$, состояниями $q_1, \ldots, q_n$ и функцией переходов δ, определяемой следующим образом. Для $\sigma \in \Sigma(\Omega)$ и $q_i, q_j \in \in \{q_1, \ldots, q_n\}$ $\delta(q_i, \sigma) = q_j$ тогда и только тогда, когда $F_i(t-1)\ \&\ f_i(t)\ \&\ \tilde{\sigma}(t)\ \&\ F_j(t) = = F_i(t-1)\ \&\ \tilde{\sigma}(t)$. Из теоремы о спецификации следует, что автомат $A(F)$, удовлетворяет спецификации F.

Построение автомата $A(F)$ состоит из двух основных этапов: 1) построения дизъюнктивного представления формулы $F(t)$, удовлетворяющего условию ортогональности, и 2) построения нормальной формы формулы $F(t)$.

На первом этапе требуемое представление получается путем последовательного применения к парам компонент дизъюнктивного представления соотношения:

$$F_i(t-1)\ \&\ f_i(t) \vee F_j(t-1)\ \&\ f_j(\mathrm{t}) \Leftrightarrow F_i(t-1)\ \&\ \neg F_j(t-1)\ \&\ f_i(t) \vee \neg F_i(t-1)\ \&\ F_j(t_1)\ \&\ f_j(t) \vee \\ \vee F_i(t-1)\ \&\ F_j(t-1)\ \&\ (f_i(t) \vee f_j(t)).$$

На втором этапе для получения нормальной формы формулы $F(t)$ осуществляется расщепление компонент ее дизъюнктивного представления путем умножения их на $\bigvee_{i=1}^{n} F_i(t)$. Эквивалентность такого преобразования формулы $F(t)$ показана в [15]. Результат расщепления компоненты определяется дизъюнктивным представлением полученного произведения, удовлетворяющим условию ортогональ-

ности. Расщепления компоненты $F_i(t-1)\&f_i(t)$ не происходит, если в таком дизъюнктивном представлении произведения все компоненты имеют левую часть $F_i(t-1)$. Процесс продолжается до тех пор, пока на очередном шаге ни одна из компонент не будет расщеплена. Описанный метод синтеза реализован в виде команды «dual» в системе синтеза и верификации дискретных устройств MVSIS, разработанной в Калифорнийском университете, в Беркли, США (http://www.ece.pdx.edu/~alanmi/mvsis/exe).

В [16] предложен метод синтеза инициального автомата, не требующий какого-либо специального представления формулы $F(t)$. Он может применяться как к д.н.ф., так и к любому другому представлению формулы. Результат синтеза получается путем индуктивного построения автомата в соответствии со структурой формулы $F(t)$, т.е. элементарные акты синтеза соответствуют основным логическим операциям, применяемым при построении формулы $F(t)$. Сходный метод синтеза используется в системе MONA [17], реализующей разрешающую процедуру для слабой теории второго порядка функции следования (WS1S). Однако там язык спецификации интерпретируется на множестве конечных слов и в качестве автомата рассматривается модель распознавателя (автомат Бюхи).

Ограничение выразительных возможностей языка спецификации средствами первого порядка, а также ограничение класса используемых формул позволили построить достаточно эффективные процедуры синтеза, практически не сокращая класса специфицируемых автоматов.

Синтез открытых систем

Для возможности корректной реализации реактивного алгоритма (представляющего собой открытую систему) одной внутренней непротиворечивости спецификации недостаточно. Непротиворечивость спецификации гарантирует только то, что существует совместное поведение внешней среды и синтезируемого алгоритма, которое удовлетворяет спецификации. Поскольку поведение среды не может быть регламентировано, необходимо, чтобы на любое допустимое ее поведение алгоритм реагировал таким образом, чтобы их совместное поведение удовлетворяло спецификации. Обычно соответствующая проблема синтеза формулируется как нахождение выигрышной стратегии в бесконечной игре между средой и системой [18]. Эта стратегия представляется в виде бесконечного размеченного дерева. Для описания всех допустимых (т.е. удовлетворяющих спецификации) стратегий строится автомат Рабина над бесконечными деревьями [19], такой, что множество всех бесконечных деревьев, распознаваемых автоматом, соответствует всем стратегиям, реализующим исходную спецификацию. Таким образом, проверка реализуемости спецификации сводится к проверке непустоты языка, представляемого автоматом Рабина. Алгоритм проверки непустоты автомата Рабина позволяет получить конечное представление бесконечного дерева, распознаваемого автоматом, которое затем преобразуется в синтезируемый алгоритм. В целом такая процедура синтеза требует времени, выражающегося в виде двойной экспоненты от размера спецификации. Приведенная оценка вряд ли может быть улучшена, поскольку, как показано в [20], проблема синтеза в рассмотренной постановке полна в классе 2EXPTIME. В [21] получено решение проблемы синтеза с существенно лучшей оценкой сложности за счет ограничения языка спецификации подмножеством GR(1) темпоральной логики LTL.

В рассматриваемом подходе к доказательному проектированию реактивных алгоритмов проблема реализуемости спецификации формулируется в терминах понятия согласованности автоматов, описывающих управляющую часть алгоритма и внешнюю для нее среду [22]. Согласованность определяется как свойство циклической композиции автоматов, задающей способ их взаимодействия.

При формализации проблемы согласованности рассматриваются частичные недетерминированные X–Y-автоматы Мура вида $A = \langle X, Y, Q, \chi, \mu \rangle$, где $\chi: Q \times X \to 2^Q$ и $\mu: Q \to Y$ – соответственно функции переходов и выходов. Если $|X| = 1$, то автомат A называется *автономным* Y-автоматом, который определяется четверкой $\langle Y, Q, \chi, \mu \rangle$. Функция переходов такого автомата имеет вид $\chi_A: Q \to 2^Q$.

Пусть $A = \langle X, Y, Q_A, \chi_A, \mu_A \rangle$ и $B = \langle Y, X, Q_B, \chi_B, \mu_B \rangle$ – частичные недетерминированные соответственно X–Y- и Y–X-автоматы Мура.

Симметричная циклическая композиция автоматов A и B представляет собой автономный недетерминированный Z-автомат Мура $C = \langle Z, Q_C, \chi_C, \mu_C \rangle$, где $Z = X \times Y \cup Y \times X$, $Q_C = Q_A \times Q_B \cup Q_B \times Q_A$, а функция переходов χ_C и функция выходов μ_C определяются следующим образом. Для всех $q \in Q_A$ и $s \in Q_B$

$\chi_C(q, s) = \{(s, q_1) \mid q_1 \in \chi_A(q, \mu_B(s))\}$,

$\chi_C(s, q) = \{(q, s_1) \mid s_1 \in \chi_B(s, \mu_A(q))\}$,

$\mu_C(q, s) = (\mu_A(q), \mu_B(s))$, $\mu_C(s, q) = (\mu_B(s), \mu_A(q))$.

Два циклических частичных детерминированных X–Y- и Y–X-автомата Мура называются *согласованными*, если их симметричная циклическая композиция имеет циклический подавтомат.

Понятие согласованности недетерминированных автоматов связано с понятием циклической детерминизации недетерминированного автомата [23].

Частичный недетерминированный циклический X–Y-автомат Мура A *согласован* с частичным недетерминированным циклическим Y–X-автоматом Мура B, если существует циклическая детерминизация автомата A, согласованная с каждой циклической детерминизацией автомата B.

Такое определение неконструктивно, поскольку количество детерминизаций автомата бесконечно, поэтому в [24] оно переформулируется в терминах конструктивного свойства *параллельной циклической композиции.* Решение проблемы рассматривается для случая, когда оба взаимодействующих автомата специфицированы множествами дизъюнктов. В основе соответствующего алгоритма лежат две процедуры: удаление из управляющей компоненты проектируемого алгоритма состояний, задаваемых формулой языка L, и проверка непротиворечивости (пополнение) множества дизъюнктов. Первая из этих процедур, так же как и вторая, реализована в виде преобразования множества дизъюнктов. Существенной особенностью метода, обеспечивающей его эффективность, является то, что не требуется рассматривать произведение взаимодействующих автоматов, т.е. объединение соответствующих множеств дизъюнктов.

Решение этой задачи состоит не только в проверке согласованности соответствующих автоматов, но и в преобразовании спецификации таким образом, чтобы все удовлетворяющие ей автоматы были согласованы со средой, после чего может применяться любой из имеющихся алгоритмов синтеза. Если же исходная спецификация проектируемого алгоритма не согласована со спецификацией среды, то реализация алгоритма не возможна.

Верификация

Хотя рассматриваемый подход к доказательному проектированию реактивных алгоритмов основан на методах синтеза, гарантирующих точное соответствие полученного алгоритма его исходной спецификации, это не исключает необходимости в ряде случаев выполнять формальную верификацию синтезированного алгоритма. Во-первых, неполнота или неточность исходной спецификации могут привести к тому, что

полученный алгоритм не будет обладать необходимым свойством, что может быть обнаружено с помощью верификации. Во-вторых, при проектировании многоагентных систем каждый из взаимодействующих между собой агентов специфицируется в отдельности, и соответствие алгоритмов функционирования агентов их спецификациям не гарантирует требуемого поведения всей системы в целом. Поэтому необходима верификация совместного поведения агентов.

Верификация формально синтезированных алгоритмов имеет свои особенности: 1) не требуется построения автоматной модели верифицируемой системы (она получается в результате синтеза) и 2) не нужно верифицировать свойства алгоритма, определяемые исходной спецификацией. В качестве языка для спецификации верифицируемых свойств алгоритма используется усовершенствованный вариант темпоральной логики с линейным временем, а верификация осуществляется методом проверки выполнимости формулы на модели (model checking) [25].

Заключение

В статье в сжатой форме излагаются основы доказательного проектирования реактивных алгоритмов исходя из их декларативной спецификации в языке логики первого порядка с одноместными предикатами. Такой подход гарантирует получение процедурного представления алгоритма, удовлетворяющего всем требованиям к его функционированию, сформулированным в виде исходной спецификации. Основная идея предложенного подхода состоит в том, чтобы определить такие ограничения на синтаксис и семантику языка спецификации, которые позволяют построить формальные методы проектирования реактивных алгоритмов промышленного уровня сложности. Возможность получения эффективных процедур проектирования обусловлена простотой выразительных средств языка L, интерпретацией языка на множестве целых, а не натуральных, чисел, спецификацией инициальных автоматов как подавтоматов неинициальных автоматов. Ограничение выразительных возможностей языка имеет не принципиальный характер, поскольку любой автомат может быть специфицирован в языке L за счет введения дополнительных предикатных символов. Интерпретация языка на множестве целых чисел делает формулы вида $\forall tF(t)$ инвариантными относительно сдвига во времени, что существенно упрощает методы проверки непротиворечивости формул. Этой же цели служит выполнение всех преобразований в процессе проектирования над неинициальными спецификациями, в которых используются атомарные формулы только вида $p(t + k)$. Это устраняет необходимость унификации в процессе резолюционного вывода.

Одной из возможных нерешенных проблем является разработка такой методологии построения исходной спецификации, которая бы облегчила написание спецификации, отражающей все необходимые требования к функционированию алгоритма, а также формулировку самих этих требований. Работа в этом направлении начата, и получены некоторые результаты, не приведенные в настоящей статье.

Литература

1. Church A. Applications of recursive arithmetic to the problem of circuit synthesis // Summaries of the summer Inst. for Symbolic Logic. – New York: Cornell Univ., 1957. – P. 3-50.
2. Büchi J.R. Weak second-order arithmetic and finite automata // Zeitschr. Math. Logik und Grundl. der Math. – 1960. – Vol. 6, № 1. – P. 66-92.
3. Трахтенброт Б.А. Синтез логических сетей, операторы которых описаны средствами исчисления одноместных предикатов // ДАН СССР. – 1958. – Т. 118, № 4. – С. 646-649.
4. Трахтенброт Б.А. Конечные автоматы и логика одноместных предикатов // Сиб. мат. журн. – 1962. – Т. 3, № 1. – С. 103-131.

5. Pnueli A. The temporal logic of programs // Proc. of the 18th Symp. on foundations of computer sci. – New York: IEEE Computer Society Press, 1977. – P. 46-57.
6. Clarke E.M., Emerson E.A. Design and synthesis of synchronization skeletons using branching time temporal logic // LNCS. – 1981. – Vol. 131. – P. 52-71.
7. Emerson E.A., Halpern J. Y. “Sometimes” and “Not Never” revisited: on branching versus linear time temporal logic // J. ACM. – 1986. – 33, № 1. – P. 157-178.
8. Stomp F.A., de Roever W.P., Gerth R.T. The μ-calculus as an assertion language for fairness arguments // Information and computation. – 1989. – Vol. 82. – P. 278-322.
9. Чеботарев А.Н. Расширение логического языка спецификации автоматов и проблема синтеза // Кибернетика и системный анализ. – 1996. – № 6. – С. 11-27.
10. Чеботарев А.Н. Об одном подходе к функциональной спецификации автоматных систем. I // Кибернетика и системный анализ. – 1993. – № 3. – С. 31-42.
11. Чеботарев А.Н. Об одном подходе к функциональной спецификации автоматных систем. II // Кибернетика и системный анализ. – 1993. – № 4. – С. 3-14.
12. Чеботарев А.Н. Проверка непротиворечивости простых спецификаций автоматных систем // Кибернетика и системный анализ. – 1994. – № 3. – С. 3-11.
13. Чеботарев А.Н. Метод раздельного резольвирования для проверки выполнимости формул языка L // Кибернетика и системный анализ. – 1998. – № 6. – С. 13-20.
14. Крывый С.Л., Чеботарев А.Н. Усовершенствованный метод проверки выполнимости множества дизъюнктов в языке L. – Таврический вестник информатики и математики. – 2006. – № 1. – С. 7-13.
15. Чеботарев А.Н. Синтез процедурного представления автомата, специфицированного в логическом языке L*. I, II // Кибернетика и системный анализ. – 1997. – № 4. – С. 60-74; № 6. – С. 115-127.
16. Капитонова Ю.В., Чеботарев А.Н. Индуктивный синтез автомата по спецификации в логическом языке L // Кибернетика и системный анализ. – 2000. – № 6. – С. 3-13.
17. Mona: monadic second-order logic in practice / J.G. Henriksen, J. Jensen, M. Jorgensen et al. // LNCS. – 1996. – Vol. 1019. – P. 89-110.
18. Abadi M., Lamport L. Composing specifications // ACM Trans. on Programming Languages and Systems. – 1993. –Vol. 15. – P. 73-132.
19. Thomas W. Automata on infinite objects // Handbook of theoretical computer science / ed. J.van Leeuwen. – Amsterdam: Elsevier Sci. Publ., 1990. – P. 134-191.
20. Pnueli A., Rosner R. On the synthesis of a reactive module // Proc. 16th Annual ACM Symp. on Principles of Programming Languages. – ACM, 1989. – P. 179-190.
21. Piterman N., Pnueli A. and Sa’ar Y. Synthesis of reactive(1) designs // Proc. of Conf. on Verification, Model Checking and Abstract Interpretation. – 2006. – P. 364-380.
22. Чеботарев А.Н. Взаимодействие автоматов // Кибернетика. – 1991. – № 6. – С. 17-29.
23. Чеботарев А.Н. Детерминизация логических спецификаций автоматов // Кибернетика и системный анализ. – 1995. – № 1. – С. 3-12.
24. Мороховец М.К., Чеботарев А.Н. Резолюционный подход к проверке согласованности взаимодействующих автоматов // Кибернетика и системный анализ. – 1994. – № 6. – С. 36-50.
25. Чеботарев А.Н. Теоретико-автоматный подход к верификации реактивных систем // Кибернетика и системный анализ. – 2001. – № 6. – С. 37-49.

А.М. Чеботарьов, А.Л. Головинський

Доказове проектування алгоритмів функціонуванняя реактивних систем

Описується підхід до доказового проектування реактивних алгоритмів, що розвивається в Інституті кібернетики ім. В.М. Глушкова НАН України. Розглядаються основні проблеми, які виникають при проектуванні реактивних алгоритмів, що специфіковані логічною мовою *L*, та методи їх розв’язання.

A.N. Chebotarev, A.L. Golovinskiy

Provably-correct Design of Algorithms of Reactive Systems Functioning

An approach to provably-correct design of reactive algorithms is described, that has been developed at the Glushkov Institute of Cybernetics of the Ukrainian Academy of Sciences. The basic problems arising in the design of reactive algorithms specified in the logical language *L*, and methods to solve them are considered.

Статья поступила в редакцию 22.07.2008.